AMENDMENTS TO THE CLAIMS

1. (PREVIOUSLY PRESENTED) In a method for implementing an event transfer system of a real time operating system kernel under a multi-tasking environment in which a priority-based preemptive scheduling is adapted, a method for implementing an event transfer system of a real time operating system kernel for a plurality of tasks in the multi-tasking environment, comprising:

each of said tasks calling, ~~by each of a plurality of tasks,~~ a kernel system function for ~~of~~ receiving an event with respect to one event under the multi-tasking environment; and

blocking ~~said~~ each of ~~the~~ said tasks and inserting ~~said~~ each of ~~the~~ said tasks into a waiting-list ~~of~~ for the event in priority order when no event is provided to the tasks, wherein all of said tasks are queued and prioritized within said waiting-list for the event;

wherein in the case that the event transfer occurs, the task having the highest priority in the waiting-list obtains the event, is ~~woken up~~ activated and an execution of said highest priority task is ~~and~~ is resumed ~~with execution~~.

2. (CURRENTLY AMENDED) The method of claim 1, wherein said waiting-list ~~of~~ for the event is managed based on the priority order so that the

task having the highest priority is arranged at the most leading portion (head) of the waiting-list.

3. (PREVIOUSLY PRESENTED) The method of claim 1, further comprising:

checking whether there is an event value already sent, when the kernel system function of receiving the event starts.

Claims 4 and 5 (CANCELLED).

6. (PREVIOUSLY PRESENTED) The method of claim 1, further comprising:

checking a validity of the event ID for thereby generating an error code in the case of validity when the kernel system function of receiving the event starts; and

returning the routine from the kernel system function.

7. (PREVIOUSLY PRESENTED) The method of claim 13, wherein when the current task is queued into the waiting-list, a time out option is additionally set if it exists.

D1

8. (CURRENTLY AMENDED) The method of claim 1, wherein when transferring the event, the method further comprises:

checking whether any waiting task exists in the waiting-list ~~of~~ for the event.

Claims 9 and 10 (CANCELLED).

11. (PREVIOUSLY PRESENTED) The method of claim 15, wherein said head task in the waiting-list which receives the event value is adjusted to a ready state and is inserted into a ready list, and an additional process routine by the sort of the event is executed.

12. (PREVIOUSLY PRESENTED) The method of claim 3, wherein as a result of the check whether the event value exists, when the event value exists, the event value is obtained an the event control block buffer, and the task routine is executed by the sort of the event.

13. (CURRENTLY AMENDED) The method of claim 3, wherein as a result of the check, when the event value does not exist, the current task is blocked and queued into the waiting-list ~~of~~ for the event.

D1

14. (PREVIOUSLY PRESENTED) The method of claim 3, wherein when the kernel system function of receiving the event starts, the method further comprises:

checking a validity of an event ID for thereby generating an error code in the case of invalidity; and

returning the routine from the kernel system function.

15. (PREVIOUSLY PRESENTED) The method of claim 8, wherein as a result of the check that whether the task exists in the waiting-list, when the waiting task does not exist, an event value is stored in an event buffer of an event control block.

16. (ORIGINAL) The method of claim 8, wherein as a result of the check whether the task exists in the waiting-list, when the waiting task exists, an event value is transferred to the head task of the waiting-list.

17. (CURRENTLY AMENDED) A method for implementing an event transfer system of a real time operating system kernel <u>for a plurality of tasks in a multi-tasking environment</u>, the method comprising:

D1

each of said tasks calling~~, by each of a plurality of tasks,~~ a kernel system function ~~of~~ for receiving an event with respect to one event in ~~a~~ the multi-tasking environment; and

blocking execution of ~~said~~ each of the tasks and inserting ~~said~~ each of the tasks into a waiting-list ~~of~~ for the event according to a priority order when no event is provided to the tasks, wherein all of said tasks are queued and prioritized within said waiting-list for the event.

18. (CURRENTLY AMENDED) The method of claim 17, further comprising:

accessing the task having the highest priority from the waiting-list ~~of~~ for the event when the event is provided; and

executing the accessed task.

19. (CURRENTLY AMENDED) The method of claim 17, further comprising:

checking whether an event value ~~exist~~ exists when the kernel system function of receiving the event starts.

D1

20. (CURRENTLY AMENDED) The method of claim 19, wherein as a result of said checking, the current task is blocked and queued into the waiting-list ~~of~~ for the event according to the priority order if the event value does not exist.

21. (PREVIOUSLY PRESENTED) The method of claim 19, further comprising:

obtaining the event value from an event control block storage and executing the task routine when said checking indicates that the event value exists.

22. (PREVIOUSLY PRESENTED) The method of claim 17, wherein when the kernel system function of receiving the event starts, the method further comprises:

checking a validity of an event ID to thereby generate an error code in the case of invalidity; and

returning the routine from the kernel system function.

23. (CURRENTLY AMENDED) The method of claim 17, further comprising:

checking whether any waiting task exists in the waiting-list ~~of~~ for the event when transferring the event; and

transferring an event value to the head task of the waiting-list when said checking indicates that the waiting task exists in the waiting-list.

24. (PREVIOUSLY PRESENTED) The method of claim 23, wherein said head task in the waiting-list which receives the event value is adjusted to a ready state and is inserted into a ready list, and an additional process routine by the sort of the event is executed.